Rechnerarchitektur
Übung 5

## Aufgabe 1

1
Was ist ein Stack im Allgemeinen? Was ist ein durch Hardware unterstützter Stack? Beschreiben Sie die grundlegende Funktionsweise und Möglichkeiten.

- ➔ Im Register ist ein Teil des Speichers als Stack aufgebaut.
- ➔ Ein Stack funktioniert, wie ein Stapel Bücher. Wenn ich ein Buch oben auf den Stapel tue, dann kann ich als nächstes nur wieder das oberste Buch vom Stapel nehmen (last-in-first-out).
- ➔ Dabei gibt es zwei Operationen:
  - PUSH = legt ein Element auf den Stapel
  - POP = entnimmt das oberste Element des Stapels
- ➔ Es kann vom Stack nicht beliebig etwas genommen werden.
- ➔ Eine weitere Möglichkeit ist, dass durch Verwendung von Stacks verschachtelte Unterprogrammaufrufe möglich sind.

2
Warum wird bei einem Unterfunktionsaufruf (call) der Inhalt des Program-Counter auf dem Stack gesichert, bei einem Sprung (jmp) aber nicht?

CALL:

- ➔ Der Stack wird verwendet, um sich die Rücksprungadresse zu merken, an die der RET-Befehl des Unterprogramms zurückkehren soll.
- ➔ Dazu legt ein Call-Befehl per PUSH die Adresse des nachfolgenden Befehls (Befehlszähler +1) auf den Stack und führt dann durch Setzen des Befehlszählers den Sprung in das Unterprogramm aus.
- ➔ Um aus dem Unterprogramm zurückzukehren, entnimmt der nächste RET-Befehl per POP die Rücksprungadresse und lädt diese in den Befehlszähler.
- ➔ Also ruft man mit Call eine „Unterroutine auf“. Diese wird abgearbeitet und dann mit RET wieder zu dem Code unter Call gesprungen und wird dort weiter gemacht.

JMP:

- ➔ Bei JMP wird das nicht gemacht, man kann nicht mit dem RET-Befehl wieder zurückspringen.

3
Globale Variablen erhalten vom C-Compiler feste Adressen im Hauptspeicher, sogenannte statische Speicherallokation. Funktionslokale Variablen werden auf dem Stack angelegt, dies wird automatische Speicherallokation genannt. Warum wird das so gemacht?

- ➔ Eine globale Variable ist außerhalb aller Funktionen definiert.
- ➔ Habe ich in meinem Hauptprogramm nun einen Block eingefügt, in dem dieselbe Variable nun mit einem anderen Wert deklariert wird; überschreibe ich vorrübergehend die globale Variable mit eben dieser lokalen Variable.
- ➔ Nach dem Block wird die ursprüngliche globale Variable wieder ausgegeben.
- ➔ Globale Variablen erhalten also eine feste Adressen im Hauptspeicher, weil sie während der gesamten Laufzeit benötigt werden.
- ➔ Wohingegen lokale Variablen nur für die Zeit belegt werden, in der der zugehörige Block aktiv ist. Deshalb sind diese auf dem Stack gespeichert.

4
Was ist ein Stackframe? Warum sind Stackframes sinnvoll? Erläutern Sie in diesem Zusammenhang die x86 Befehle ENTER und LEAVE. Was muss beachtet werden, wenn sowohl ENTER und LEAVE als auch PUSH und POP benutzt werden sollen?

- ➔ Stackframes sind eine Speicherverwaltungstechnik in einiger Programmiersprachen zum Erzeugen und Eliminieren temporärer Variablen. Sie sind sinnvoll, weil sie den rekursiven Aufruf von Unterprogrammen erlauben.
- ➔ Der „Stack Frame“ ist wie ein Rahmen, der um einen Teil des Stacks gezogen wird. Dieser Rahmen enthält alle Parameter, die lokalen Variablen und die Rücksprungadresse.
- ➔ ENTER erzeugt einen Stackframe
- ➔ LEAVE eliminiert einen Stackframe
- ➔ PUSH speichert einen Wert auf dem Stack
- ➔ POP ruft den zuletzt auf den Stack geschobenen Wert ab

## Aufgabe 2

Warum wird Rekursion benötigt?
Eine Funktion, die rekursiv definiert ist, hat eine längere Laufzeit als eine Funktion, die iterativ definiert ist. Eine rekursive Funktion ruft sich selbst auf und die iterative Funktion nicht. Aber Rekursion wird noch gebraucht, weil einige Funktionen keine einfache geschlossene Form haben oder es nicht möglich ist eine direkte Form der Funktion zu finden.